REMARKS

Applicants have studied the Office Action dated December 29, 2008, and have made amendments to the claims. It is submitted that the application, as amended, is in condition for allowance. By virtue of this amendment, claims 1, 2, 5-8, and 21-33 are pending. Claim 4 has been canceled without prejudice. Claim 1 has been amended and new claims 25-33 have been added. Reconsideration and allowance of the pending claims in view of the above amendments and the following remarks are respectfully requested.

Claims 1, 2, 4-8, and 21-24 were rejected under 35 U.S.C. § 101 for being directed to non-statutory subject matter. Claim 4 has been canceled so, with respect to this claim, this rejection is moot. With respect to claims 1, 2, 5-8, and 21-24, this rejection is respectfully traversed. Applicants have amended claim 1 to recite the steps of "storing, into a computer system," "registering, by the computer system," and "directly notifying, by the computer system". Applicants submit that a method reciting such steps being performed by and into a computer system is statutory subject matter. Therefore, it is respectfully submitted that the rejection of claims 1, 2, 5-8, and 21-24 under 35 U.S.C. § 101 should be withdrawn.

Claims 1, 2, 4-8, and 21-24 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Hutsch et al. (U.S. Patent Application Publication No. 2001/0034771) in view of Krishnaswami et al. (U.S. Patent Application Publication No. 2005/0091346). Claim 4 has been canceled so, with respect to this claim, this rejection is moot. With respect to claims 1, 2, 5-8, and 21-24, this rejection is respectfully traversed.

The present invention is directed to an efficient and easy-to-implement method for managing configuration data. One embodiment of the present invention provides a method for managing configuration data. According to the method, configuration values are stored in a hierarchical tree having multiple nodes, a defined structure, and defined data types for the stored configuration values. Each node is associated with at least one of the configuration values, and each of the

configuration values dictates how an application component associated with that configuration value behaves and/or interacts with other application components. An application component is an entity that is defined, in part, by the at least one configuration value.

Additionally, some of the nodes are only associated with a set of configuration values, while other of the nodes are associated with a combination of a set of configuration values and an identifier associated with at least one application component. An application component is registered with at least one of the nodes of the tree, based on a query received from the application component. The at least one application component is an entity that is defined, in part, by the at least one configuration value. The application component is directly notified when a configuration value associated with the at least one node of the tree is modified, based on an addition or change in at least one configuration value that matches the query. The notification comprises at least the configuration value that has been modified.

For example, in the exemplary embodiment disclosed in the specification, configuration data, which includes configuration values, is organized into a hierarchical tree. Application components register interest in a particular node of the tree, or a particular sub-tree. The registered interest is stored within the node. Because they registered, these components receive notifications whenever a corresponding configuration value changes. Thus, any component that is a current holder of a registration or reference will be notified when a configuration value changes. Because the configuration values are arranged hierarchically in a tree such as an XML tree, the various application components can register for callbacks or notification upon modification of any nodes in any sub-tree.

Further, the structure of the configuration data in the XML tree can be altered, for example by adding more sub-nodes to a particular node in the tree. An interested party to the changed sub-tree will receive the new configuration data. Thus, an application component can register itself as an interested party to any change in a configuration value. Also, the use of XML to represent the tree allows a sub-tree of configuration data to be easily be expanded, with no change in how an

application component handles those configuration values. Thus, an application component (such as a Graphical User Interface, an object, an Application Programming Interface, a plug-in, an application, or a user) has the ability to handle changing configuration data, and can handle configuration data in a hierarchical structure such as by using the extensibility of XML.

The Hutsch reference is directed toward a network portal system that allows universal and integral use of different services by arbitrary client systems. The network portal system links, via a communication network, multiple content provider systems with multiple client systems. The network portal system allows a client to get content from a provider system even if the client system and the provider system do not use the same communication protocol. In the system of Hutsch, a web-top manager within the portal system receives a content request from a client system. The web-top manager communicates with a universal content broker system that also is in the portal system. Upon receipt of a content request from the web-top manager, a universal content broker, selects a content provider system that is able to provide the requested content. The universal content broker accesses the selected content provider system, and issues a request that results in the performance of the action specified in the request by the accessed provider system.

Additionally, in the system of Hutsch, if the request was to retrieve content, the content in a raw data format is passed to the web-top manager. The web-top manager renders the requested content into a page that can be displayed by the requesting client system. This page is then returned to the requesting client system. Further, the universal content broker system of Hutsch can include a configuration server and a configuration proxy that is coupled to the configuration server. The configuration server includes a first DOM tree that includes user profiles and application profiles. The Configuration proxy includes a second DOM tree that includes a subset of data in the first DOM tree.

The Krishnaswami reference is directed to a system and method for facilitating configuration management. The system includes a configuration store that stores persisted configuration and/or dependency information associated with one or more applications, and a configuration service

component that manages access to the configuration store. The system can further include a configuration management engine (e.g., API) that allows a client application to access, query, and/or modify one or more settings. An application can submit an XML assembly manifest which comprises the assembly identity, the application binaries, its dependencies, etc. The manifest can also include a configuration section that declaratively specifies the persisted settings for the application. The configuration section includes an XSD-based schema that defines rich types for the settings and the settings themselves, and metadata for these settings including description and default values, manageability attributes (e.g., migrate, backup, policy), and integrity constraints called assertions (that could potentially describe the relationships between settings).

Amended claim 1 recites:

> storing, into a computer system, a plurality of configuration values in a hierarchical tree having a plurality of nodes, a defined structure, and defined data types for the stored configuration values, wherein the plurality of nodes includes at least one inner node and at least one child node that is associated with the inner node, wherein at least one configuration value is stored in each node of the plurality of nodes, and each of the configuration values dictates how an application component associated with that configuration value at least one of behaves and interacts with other application components, wherein an application component is an entity that is defined, in part, by the at least one configuration value, and wherein some of the nodes store a set of configuration values while other of the nodes store a combination of a set of configuration values and an identifier associated with at least one application component;
>
> registering, by the computer system, at least one application component directly with at least one of the nodes of the tree, based on at least one query received from the at least one application component, wherein the at least one node comprises at least one configuration value that dictates how the application component at least one of behaves with and interacts with other application components, and wherein the at least one application component is an entity that is defined, in part, by the at least one configuration value; and
>
> directly notifying, by the computer system, the at least one application component when a configuration value associated with stored in the at least one node is modified, based on an addition or change in at least one configuration value that matches the at least one query, wherein the notification comprises at least the configuration value that has been modified..
>
> .

As explained in the previous Amendment, in embodiments of the present invention, some of the nodes are only associated with a set of configuration values, and other of the nodes are associated with a combination of a set of configuration values and an identifier associated with at least one application component. With respect to this, the Examiner stated that Hutsch teaches:

> wherein some of the nodes are only associated with a set of configuration values while other of the nodes are associated with a combination of a set of configuration values [value] and an identifier [key] associated with at least one application component (see [0158] and [0159]);

The "value" and "key" of Hutsch cited by the Examiner refer to a key-value pair. Paragraphs 0158 and 0159 of Hutsch merely state:

> [0158] All of the entries are stored in hierarchical form in key-value pairs. As explained more completely, below a configuration tree contains separate branches for this purpose, which can be used for different users or the various user devices. The configuration tree is described through a Document Object Model (DOM) based on XML (Extended Markup Language). Communication between configuration server 336 and the various components of network portal system 100 is provided via requests for entries or requests for modification of entries. In one embodiment, the data is exchanged using an XML-based protocol.
>
> [0159] Alterations to the DOM tree are carried out by transactions, which include inserting new nodes or new key-value pairs describing user preferences. This also covers such things as modification of individual entries or deletion of an entire subtree as the entries are no longer needed. Configuration server 336 also contains different mechanisms for querying the status of transactions or registering different listeners that are notified when alterations are made to configuration service elements and that in turn notify the applications affected.

Thus, the key-value pairs only describe user preferences. Nowhere do paragraphs 0158 and 0159 of Hutsch teach or suggest that some of the nodes store a set of configuration values while other of the nodes store a combination of a set of configuration values and an identifier associated with at least one application component. In other words, Hutsch clearly does not teach both: (1) some nodes store a set of configuration values, and (2) other nodes store a combination of configuration values and an identifier associated with at least one application component. At best, Hutsch teaches that all nodes include key-value pairs, which the Examiner is equating to a configuration value and an identifier. This teaches away from the recited feature of the present

invention that some of the nodes store a set of configuration values while other of the nodes store a combination of a set of configuration values and an identifier associated with at least one application component. Accordingly, the present invention distinguishes over Hutsch for at least this reason.

Additionally, in embodiments of the present invention the application is directly registered with the node itself. Stated differently, the reference to an application component that is to be notified when a change occurs is stored within the node itself. Hutsch teaches that listeners can register to be notified when alterations are made to configuration service elements. These listeners then notify the applications that are affected. With respect to this, the Examiner stated that Hutsch teaches:

> registering at least one application component with at least one of the nodes of the tree, based on at least one query [transaction] received from the at least one application component (see [0159])

The transaction that paragraph 0159 of Hutsch teaches is the transaction that actually **alters** the DOM tree. This is in no way the same as a "query received from the at least one application component". Paragraph 0159 does not teach or suggest that an application that wants to be registered with a node submits a query that is received. Hutsch merely teaches that the DOM tree is altered based on transactions. Hutsch further teaches that different mechanisms for querying the states of transaction or registering different listeners to be notified when alterations are made to configuration service elements. This clearly shows that the transactions that the Examiner is equating to the presently claimed "query" are separate and distinct from the mechanism used to register a listener. Hutsch does not register a listener using transactions that alter the DOM tree. Accordingly, the present invention distinguishes over Hutsch for at least this reason as well.

Furthermore, the Examiner continues to equate Hutsch's "listener" with the recited "application component". This is a misplaced comparison. The listener of Hutsch is not an application component that has operations and behaviors that are dictated by configuration values. A listener merely notifies an application component of alterations of a service component. It is improper for the Examiner to perform a piecemeal dissection of the claims and assert that Hutsch teaches an application component as recited for the present invention when the present invention

recites more than just an "application component" or a "configuration vale". The recited application components of the present invention have operations and behaviors that are dictated by the recited configuration values. The Examiner admits that Hutsch does not teach configuration values that dictate how an application component associated with that configuration value at least one of behaves and interacts with other application component, and goes on to combine Krishnaswami with Hutsch to overcome this deficiency. Hutsch does not teach an "application component" and a "configuration value" as recited in the present claims.

Additionally, amended claim 1 now recites that "the at least one application component is an entity that is defined, in part, by the at least one configuration value". Hutsch clearly does not teach or suggest this recites feature because the listener of Hutsch is not defined, in part, by the at least one configuration value. Accordingly, the present invention distinguishes over Hutsch for at least these reasons as well.

The Examiner also states that Hutsch teaches:

> directly notifying the at least one application component [listener] when a configuration value stored in the at least one node is modified [alterations], based on an addition or change in at least one configuration value that matches the at least one query [transaction] (see [0159]).

However, as has been repeatedly discussed, the Examiner's comparison of a "listener" to the recited "application component" is incorrect and improper. Also, amended claim 1 now recites that "the notification comprises at least the configuration value that has been modified". Hutsch only teaches that the listener is notified that a configuration service element has been modified and does not teach or suggest that the actual modification is sent to the listener or the application associated with the listener. Accordingly, the present invention distinguishes over Hutsch for at least this reason as well.

The Examiner correctly states that Hutsch "fails to explicitly disclose the further limitations of wherein at least one configuration value is stored in each node of the plurality of nodes and

wherein the at least one node comprises at least one configuration value that dictates how the application component at least one of behaves with and interacts with other application components." However, the Examiner goes on to combine Krishnaswami with Hutsch stating that Krishnaswami discloses:

> a configuration manager, including the further limitations of wherein at least one configuration value is stored in each node of the plurality of nodes (see [0114]); registering at least one application component directly with at least one of the nodes of the tree (see [0320]),wherein the at least one node comprises at least one configuration value that dictates how the application component at least one of behaves with and interacts with other application components [dependencies] (see [0039] and [0063]); and directly notifying the at least one application component when a configuration value is modified [notification of that change is sent to the application] (see [0320]).

However, paragraphs 0114 and 0320 of Krishnaswami merely state:

> [0114] Turning to FIG. 6, a block diagram of a configuration management system architecture 600 in accordance with an aspect of the present invention is illustrated. The architecture 600 is implemented in five functional layers of components: the Server-Side Virtual Document Layer 610, the Intermediate Handler Layer 620, the Handler Layer 630, the Assertions Engine 640, and the Notifications agent (not shown). The Server-Side Virtual Document Layer 610 serves to provide the functionality as does the Client-Side Virtual Document Layer; however, in accomplishing this end it presents the Configuration management engine 250(s) (e.g., configuration management engine(s) 250) with a shared view of the settings namespace, thereby allowing multiple reads to occur against the same resource. This has the effect of conserving the system's memory reserves, a much valued optimization. If not already in shared memory, a namespace open in Read-Committed Mode triggers a Virtual Document Layer view of the namespace to be loaded. A namespace save will first reconcile the settings changes in the data-store and then reflect those changes in the shared memory view.
>
> [0320] Client applications can also be interested in listening for modifications to their settings and metadata from external sources, such as other processes or management service(s) 260. This can be accomplished through a mechanism known as notification. If an application has registered for a notification on a particular namespace, then whenever a change is committed to that namespace, a notification (of that change) is sent to the application. The notification invokes a special handler, specified by the application in the registration process. Applications can call the RegisterForNotification( ) method from within the SettingsNamespace instance to register for a notification on that namespace. Applications can also call the

UnregisterForNotification( ) method from within the SettingsNamespace instance to stop receiving notifications of changes on that namespace.

These sections of Krishnaswami do not teach or suggest a hierarchical tree having a plurality of nodes, that at least one configuration value is stored in each node of the plurality of nodes, and that each of the configuration values dictates how an application component associated with that configuration value at least one of behaves and interacts with other application components. Krishnaswami merely teaches a namespace and does not teach or suggest that this namespace is a hierarchical tree having a plurality of nodes and that at least one configuration value is stored in each node of the plurality of nodes. Accordingly, the present invention distinguishes over Hutsch and Krishnaswami alone and/or in combination for at least these reasons.

Furthermore, Krishnaswami explicitly states that a notification is sent to registered clients when settings in a namespace have changed and that this notification has a transaction ID that the clients use to query for specific changes in the namespace. See Krishnaswami at paragraphs 0123 and 0251. Krishnaswami does not teach or suggest that the notification comprises at least the configuration value that has been modified. Accordingly, the present invention distinguishes over Hutsch and Krishnaswami alone and/or in combination for at least this reason as well.

Applicants believe that the differences between Hutsch, Krishnaswami, and the present invention are clear in amended claim 1, which sets forth one embodiment of the present invention. Therefore, claim 1 distinguishes over the Hutsch and Krishnaswami references, and the rejection of this claims under 35 U.S.C. § 103(a) should be withdrawn.

As discussed above, amended claim 1 distinguishes over the Hutsch and Krishnaswami references, and thus, claims 2, 5-8, and 21-24 (which depend from claim 1) also distinguish over the Hutsch and Krishnaswami references.

Further, it is submitted that limitations recited in the dependent claims are not taught by Hutsch and Krishnaswami. For example, with respect to dependent claims 21-24, the Examiner directs Applicants to paragraph 0159 of Hutsch. However, paragraph 0159 of Hutsch is completely

irrelevant to the recited feature of the plurality of configuration values in the hierarchical tree including all of the configuration data values that are required by the at least one application component, as is recited in claim 21. Paragraph 0159 of Hutsch is also completely irrelevant to the recited feature of registering the at least one application component directly with the at least one inner node, as is recited in claim 22. Also, paragraph 0159 of Hutsch is completely irrelevant to the recited feature of directly notifying the at least one application component when at least one configuration value associated with at least one of the inner node and the child node associated with the inner node is modified, based on an addition or change in the at least one configuration value, as is recited in claim 23. Hutsch explicitly states that data only exists at leaf nodes of a tree. Therefore, Hutsch cannot possibly teach that a configuration value is stored within an inner node. Paragraph 0159 of Hutsch is further irrelevant to the recited feature of at least one configuration value in the plurality of configuration values that is associated with a first application component overlapping with another configuration value in the plurality of configuration values that is associated with a second application component, the at least one configuration value and the another configuration value being nested under a common sub-tree in the tree, as is recited in claim 24. Hutsch does not even mention the overlapping of values in paragraph 0159. Applicant request that the Examiner explain how paragraph 0159 of Hutsch teaches or suggest each of the se recited features.

Therefore, it is respectfully submitted that the rejection of claims 1, 2, 5-8, and 21-24 under 35 U.S.C. § 103(a) should be withdrawn.

Claims 25-33 have been added by this amendment, and are provided to further define the invention disclosed in the specification. Claims 25-33 are allowable for at least the reasons set forth above with respect to claims 1, 2, 5-8, and 21-24.

Further, new claims 26 and 32 correspond to previously canceled claims 9 and 17, respectively. New claim 25 recites:

> storing at least one other configuration value in the at least one nodes that after the at least one application component has been directly registered with at least one of the nodes;
>
> wherein directly notifying the at least one application component further comprises the step of:

directly notifying the at least one application component when both the at least one configuration value and the at least one other configuration value are modified.

Nowhere do Hutsch and Krishnaswami alone and/or in combination teach or suggest these recited features. Hutsch only teaches that a listener registers for changes in elements that exist at the time of the listener registration. Krishnaswami only teaches a namespace comprising settings. Nowhere does Krishnaswami teach that settings can be added to the namespace after the client registers for change notifications and that the client is notified of changes to these new settings.

No amendment made was related to the statutory requirements of patentability unless expressly stated herein. No amendment made was for the purpose of narrowing the scope of any claim, unless Applicants have argued herein that such amendment was made to distinguish over a particular reference or combination of references.

Applicants have examined the references cited by the Examiner as pertinent but not relied upon. It is believed that these references neither disclose nor make obvious the invention recited in the present claims. In view of the foregoing, it is respectfully submitted that the application and the claims are in condition for allowance. Reexamination and reconsideration of the application, as amended, are requested.

If for any reason the Examiner finds the application other than in condition for allowance, the Examiner is invited to call the undersigned attorney at (561) 989-9811 should the Examiner believe a telephone interview would advance the prosecution of the application.

Respectfully submitted,

Date: March 30, 2009

By: /Stephen Bongini/
Stephen Bongini
Reg. No. 40,917
Attorney for Applicants

FLEIT GIBBONS GUTMAN
BONGINI & BIANCO P.L.
One Boca Commerce Center
551 Northwest 77th Street, Suite 111
Boca Raton, Florida 33487
Telephone: (561) 989-9811
Facsimile: (561) 989-9812